Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM APPLE UK101 ORANGE

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.3 No.3 December 1982

Merry Christmas and a Happy New Year

INDEX

Assembler Enhancements	9	PICO DOS Auto Boot	8
Bench Marks	16	Proton	16
Dear Paul	11	Rabble 6502	6
Fast Access Cartridge	4	Simple EPROM Programmer	7
For Sale or Swap	16	Software Review	5
Front Page	3	Superboard	4
Letter from Earl Morris	3	The Meeting was KAOS	10
Machine Language pt 6	13	TV Typewriter	2
Modems	7	User Friendly Programming	15
Oops	16		

TELEVISION TYPEWRITER
by David Dodds

I have two young boys who love to pound the keys of my Superboard. To keep them amused (and protect whatever is in the machine), I make use of a very simple television typewriter routine in machine language.
The routine is:-

```
          START     JSR $FEED   ; get a character from keyboard
                    JSR $BF2D   ; put it on the screen
                    CLC
                    BCC START   ; branch always
```

You can enter this routine, in seconds, anywhere you like as it is completely relocatable. For example to enter it at $0222, use the monitor and type   0222/20*ED*FE*20*2D*BF*18*90*F7.0222G   Where * is carriage return.

I was playing around with this one day and discovered that many of the machines graphics characters can be displayed by using the CTRL key in conjunction with other keys. Carriage return and line feed keys still act as normal so I started composing pictures.

The program below will introduce you to a few of the new friends I made while fooling around. While the program doesn't do a lot it does give some idea of what can be produced with OSI graphics. I have created over 20 cousins to the 'critters' in this program.

```
LIST

 1 POKE251,0:GOTO57
 2 REM 3 PART GRAPHICS DISPLAY ROUTINE
 4 FORY=1TO24:POKEHA+Y,ASC(MID$(HA$(I),Y,1))
 6 POKEBO+Y,ASC(MID$(BO$(I),Y,1))
 8 POKEFE+Y,ASC(MID$(FE$(I,K),Y,1)):NEXT:RETURN
 9 I=0:K=1:GOSUB4:FORI=1TO4
 10 FORK=4TO1STEP-1:GOSUB4:NEXTK,I
 11 FORI=3TO0STEP-1:FORK=1TO4:GOSUB4:NEXTK,I:GOT
O9
 12 REM DATA FOR GRAPHICS:NOTE REPITITIVE NATURE
 OF DATA
 13 REM DABUG CAN EASE TYPING IF YOU ARE SMART
 14 DATA 32,32,32,32,5,8,32,32,32,32,243,246,32,
32,32,32,6,7
 15 DATA 32,32,32,32,19,21
 20 DATA 32,32,32,32,242,245,32,32,32,32,242,245
,32,32,32,32,242,245
 25 DATA 32,32,32,32,242,245
 30 DATA 32,32,32,32,242,242,32,32,32,32,242,242
,32,32,32,32,242,242
 35 DATA 32,32,32,32,242,242
 40 DATA 32,32,32,32,245,242,32,32,32,32,245,242
,32,32,32,32,245,242
 45 DATA 32,32,32,32,245,242
 46 GOTO55
 50 DATA 32,32,32,32,245,245,32,32,32,32,245,245
,32,32,32,32,245,245
 55 DATA 32,32,32,32,245,245
 57 PRINTCHR$(127):REM DABUG SCREENCLEAR
 58 LL=32:SCREEN=53572:IFPEEK(65506)=1THENLL=64:
SCREEN=53844
 59 DIMFE$(4,4):HAT=SCREEN:BODY=HAT+LL:FEET=BODY
+LL
 60 FORJ=1TO24:READC:HA$(4)=HA$(4)+CHR$(C):NEXT
 61 FORJ=1TO24:READC:BO$(4)=BO$(4)+CHR$(C):NEXT
 62 FORI=1TO3:FORJ=1TO24:READC:F$(I)=F$(I)+CHR$(
C):NEXTJ,I
 63 FORI=3TO0STEP-1:HA$(I)=RIGHT$(HA$(I+1),23)+"
 "
 64 BO$(I)=RIGHT$(BO$(I+1),23)+" ":NEXT
 65 FORJ=1TO3:FE$(4,J)=F$(J):NEXT
 66 FE$(4,4)=FE$(4,2)
 67 FORI=3TO0STEP-1:FORJ=4TO1STEP-1:FE$(I,J)=RIG
HT$(FE$(I+1,J),23)+" "
 68 NEXTJ,I:PRINT"   MARTIAN TWO STEP":PRINT:PRI
NT:GOTO9
OK
```

All manner of things like buildings, flowers, border patterns and fancy characters can be created as well. Try it for yourself. You never know what you will discover. Be wary of 'CNTRL J' and 'CNTRL M'.

TRANSFERRED FROM FRONT PAGE

We thought we would save ourselves some time and energy by putting Christmas greetings on the front page, but there are a few odds and ends we think you should know.

When writing or ringing COMP-SOFT for prices on goods, please remember to give your name and state that you are a member of KAOS, otherwise you will be quoted and charged the normal price. As a matter of interest, COMP-SOFT will be closing on 23rd December 1982 and re-opening on 17th January 1983. Let's hope George doesn't become a member of the Greek army.

Would SYM members please note that Brian Campbell has a new phone number 03 211 4793.

For those people who did not read the front page last month, we repeat - there will not be a meeting in December. The closing date for articles for the next newsletter will be the 14th January 1983.

A LETTER FROM EARL MORRIS

Dear Kaos,

The last several issues of your newsletter had questions and comments on the now famous OSI Garbage Collector Bug. You have quoted that the code from PEEK appears to solve the problem. I have heard no complaints. Enclosed is a hex dump of the code if any of your readers want to put it into EPROM.

Some people have been having problems replacing ROMs with EPROM on the Superboard. OSI has connected phase 2 clock to pin 21. Some brands of 2716 object to this treatment. Most will work with a system clock of one meg, but fail at faster CPU rates. Even some 2716-1's will not work properly. The solution, as pointed out by David Jones in the August 1982 issue of PEEK, is to connect pin 21 of the EPROM to 5 volts. So if you have had trouble replacing OSI's ROM, try this fix.

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
B140   A9 80 85 60 68 D0 D0 A6 85 A5 86 86 81 85 82 A0
B150   00 84 9D A5 7F A6 80 85 AA 86 AB A9 68 85 71 84
B160   72 C5 65 F0 05 20 D9 B1 F0 F7 A9 06 85 A0 A5 7B
B170   A6 7C 85 71 86 72 E4 7E D0 04 C5 7D F0 05 20 D3
B180   B1 F0 F3 85 A4 86 A5 A9 04 85 A0 A5 A4 A6 A5 E4
B190   80 D0 07 C5 7F D0 03 4C 18 B2 85 71 86 72 A0 01
B1A0   B1 71 08 C8 B1 71 65 A4 85 A4 C8 B1 71 65 A5 85
B1B0   A5 28 10 D7 C8 B1 71 A0 00 0A 69 05 65 71 85 71
B1C0   90 02 E6 72 A6 72 E4 A5 D0 04 C5 A4 F0 C1 20 D9
B1D0   B1 F0 F3 C8 B1 71 10 30 C8 B1 71 F0 2B C8 B1 71
B1E0   AA C8 B1 71 C5 82 90 06 D0 1E E4 81 B0 1A C5 AB
B1F0   90 16 D0 04 E4 AA 90 10 86 AA 85 AB A5 71 A6 72
B200   85 9C 86 9D 88 88 84 A2 A5 A0 18 65 71 85 71 90
B210   02 E6 72 A6 72 A0 00 60 C6 A0 A6 9D F0 F5 A4 A2
B220   18 B1 9C 65 AA 85 A6 A5 AB 69 00 85 A7 A5 81 A6
B230   82 85 A4 86 A5 20 D6 A1 A4 A2 C8 A5 A4 91 9C AA
B240   E6 A5 A5 A5 C8 91 9C 4C 4B B1 EA EA EA
```

Superboard

NEWSLETTER OF THE OHIO SUPERBOARD USER GROUP, 146 YORK STREET, NUNDAH, 4012.

A FAST ACCESS CARTRIDGE SYSTEM by Bruce Dykstra.

The aim of this project was to provide at minimal cost, a system to access any program on a tape within 20 seconds under software control.

The recorder used is a standard 8 track car player minus the audio electronics. The one actually used was an AWA Clarion transport which cost $28. Tapes are available from Tandy, and they can always be rewound with high quality reel-to-reel tape for better performance.

Basically, the operation of the system is as follows.

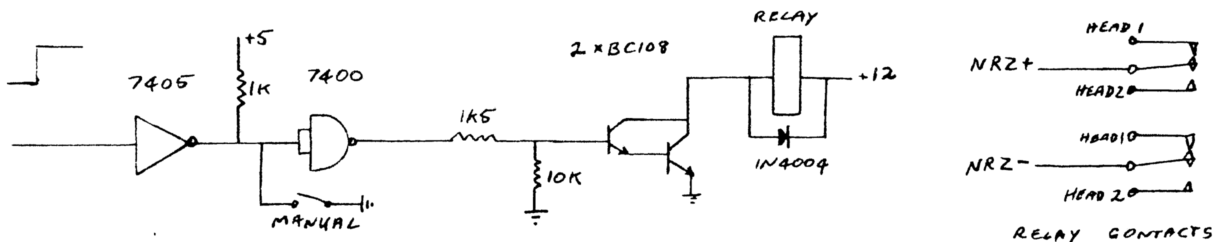
Track 1,head 1 holds the program to run the system, and also contains an index of what is on the tape and at which locations. Using a VIA for sensing, the computer has inputs from track and head selectors to find the correct track. A counter counts each rotation of the flywheel.

Say the program you want is on track 3, head 2, at 200 counts. The computer program peeks and pokes the track select until it is on the correct track, selects head 2, and turns on the fast forward motor. When the count reaches 190, the fast forward motor is switched off, and the play motor is turned on. The correct program then loads when the filename is found.

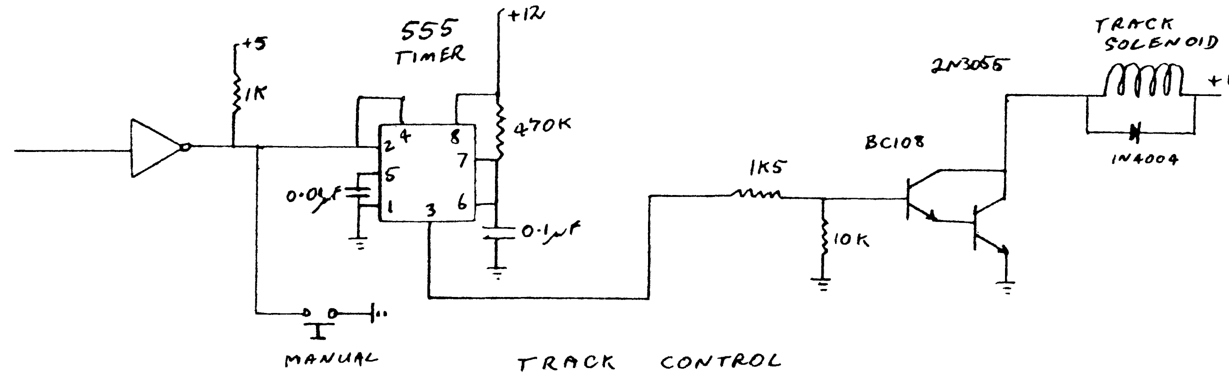
The recording technique used is NRZ, which makes for simple circuitry and works fine at 1200 Baud and may even go higher.

Having loaded the program, the tape mechanism goes to fast forward back to the start of the tape, resets the counter to zero, and stops, ready for the next use.

Following are some of the circuits used in the device. More information is available from the User Group or the author at 27 Smith St, Ulverstone. 7315.

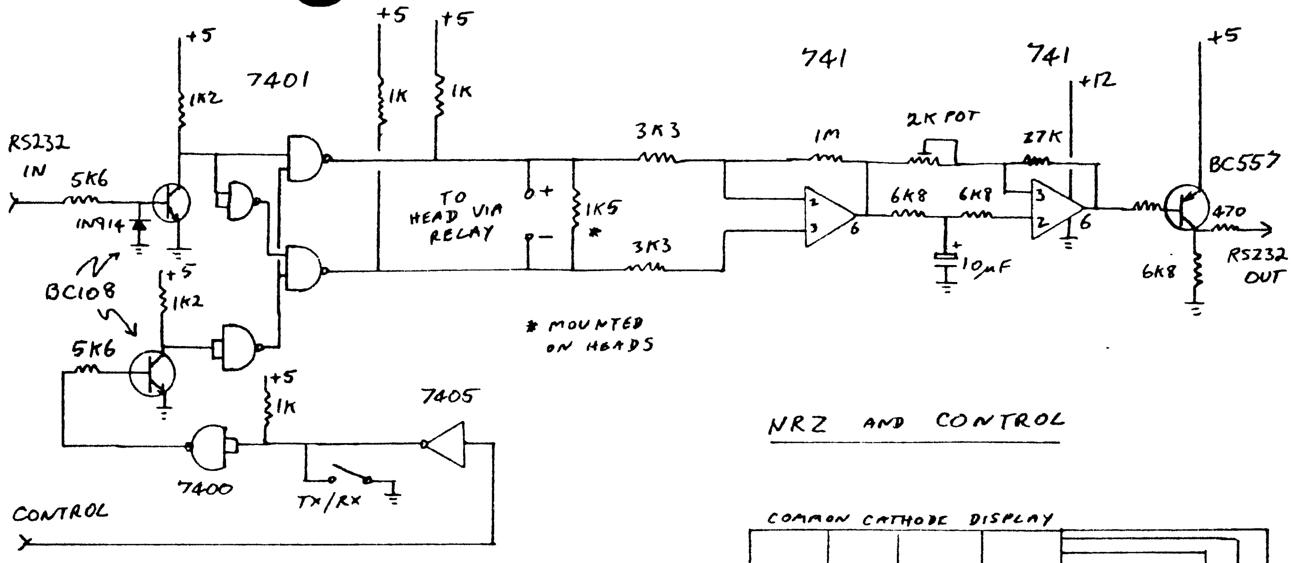


HEAD CONTROL

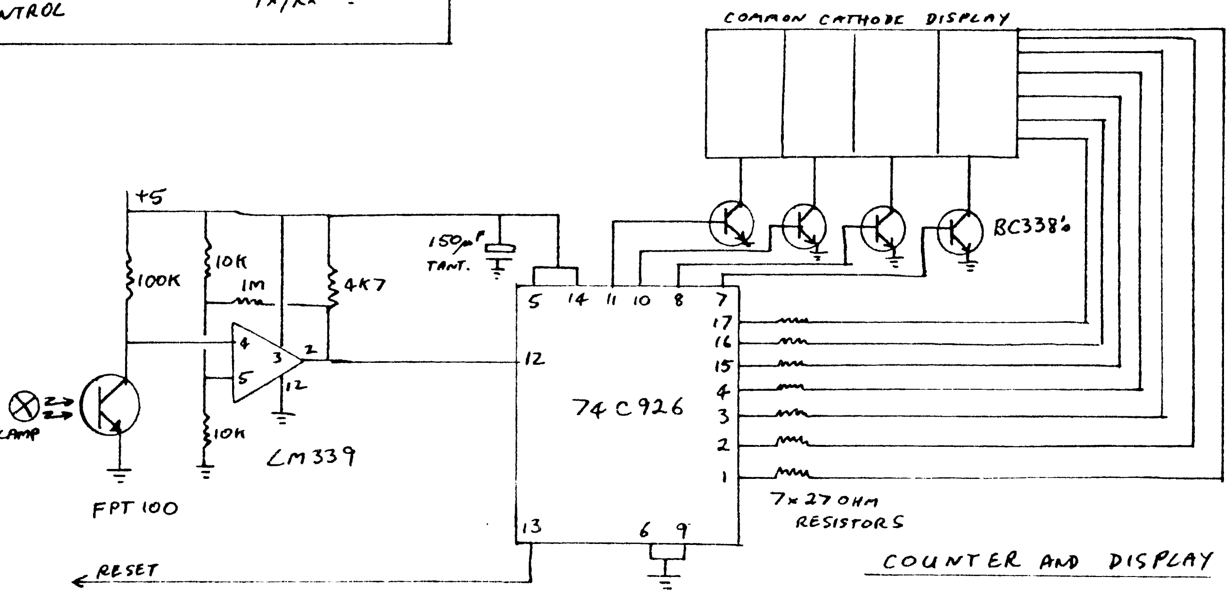


TRACK CONTROL

Superboard



NRZ AND CONTROL



COUNTER AND DISPLAY

SOFTWARE REVIEW - *Personal Calendar.*

Personal Calendar is a program requiring 8k. It comes on the OSI label and is written by David Tewksbary. The program itself occupies 4k, leaving 3k space for appointments. On running, a menu appears. Options are Find, Add to, or Review appointments. There are three data fields, Date, Time and appointment details. You can also delete appointments, and save future appointments to tape as data statements. The program does it all neatly and efficiently, but the processes are simple, and anyone who has fiddled with data storage and retrieval would have no problems writing their own routines.
As with all Utility programs, one has to ask how useful it really is. I think on reflection, that I would prefer a shapely secretary to keep my appointments in order. Personal Calendar is in the OSUG library at the usual postage rates.

Merry Xmas and a Happy New Year,
Ed Richardson.

THE RABBLE 65

DESIGN CRITERIA, to produce a single board, 6502 based, OSI compatable computer as a companion board to the RABBLE Expansion board. Features to include: integral keyboard, FDC, modem and 6845 video. The following information indicates a configuration that is similar to the Superboard or C1.

NORMAL CONFIGURATION

16K user RAM	0000 - 3FFF	6116 (eight)	
2K user RAM	C800 - CFFF	6116	
2K video RAM	D000 - D7FF	6116	
128 bytes system RAM at C400 - C47F			
8K ROM	A000 - BFFF	2732	(intend for BASIC or FORTH)
8K ROM	E000 - FFFF	2732	(lower 4K user ROM, monitor extensions etc. Locations F000-F001 and FC00-FC01 are decoded independently for serial I/O. The upper 4K will be arranged so that main subroutine entry points are maintained to give maximum software compatibility.)

MONITOR EXTENSIONS will include;

1. Disassembler
2. GTBUG screen driver/editor (by Tony Durrant)
3. Trace facilities

MAIN FEATURES

Integral keyboard which may be seperated and used remotely via a 16 way ribbon cable

6845 CRT controller (later versions may use the 6545)

Standard display formats will be 32X32, 64X32, 80X24 (option for those with good quality monitors)

Included on the board is an OSI compatable Floppy disk controller using the SMC 9216 data seperator, the connecting cable to the drive is via a 50 way connector which can be strapped to suit the normal 8" drives or the MPI 5.25" drives.

SERIAL I/O, on board 6850 ACIA for cassette and printer via RS232 port with selectable baud rates from MC14411 baud rate generator.

By moving the screen driver to block F, we have opened up a page at the top of BASIC in ROM, extensions so far are;

CLS (clear screen)
TRON,TROF (trace on/off)
RENUMBER (renumbers lines in BASIC including GOSUB and GOTO)
AUTO (auto increment for line numbers)

The system will be supplied with three manuals;

1. HARDWARE (including all circuit diagrams)
2. SYSTEM SOFTWARE (including source for monitor)
3. APPLICATION SOFTWARE (documentation for ROM BASIC, FORTH etc.)

Expected costs, a minimum system should cost approximately $300 assembled. Built as a minimum disk system, together with the RABBLE expansion board, and including a 5.25" drive, power supplies etc. the cost should be less than $1000. The first productions versions should be on display at the February KAOS meeting.

Ray Gardiner and Bill Chilcott would welcome any suggestions or queries from KAOS members....

MODEMS
by Ed Richardson

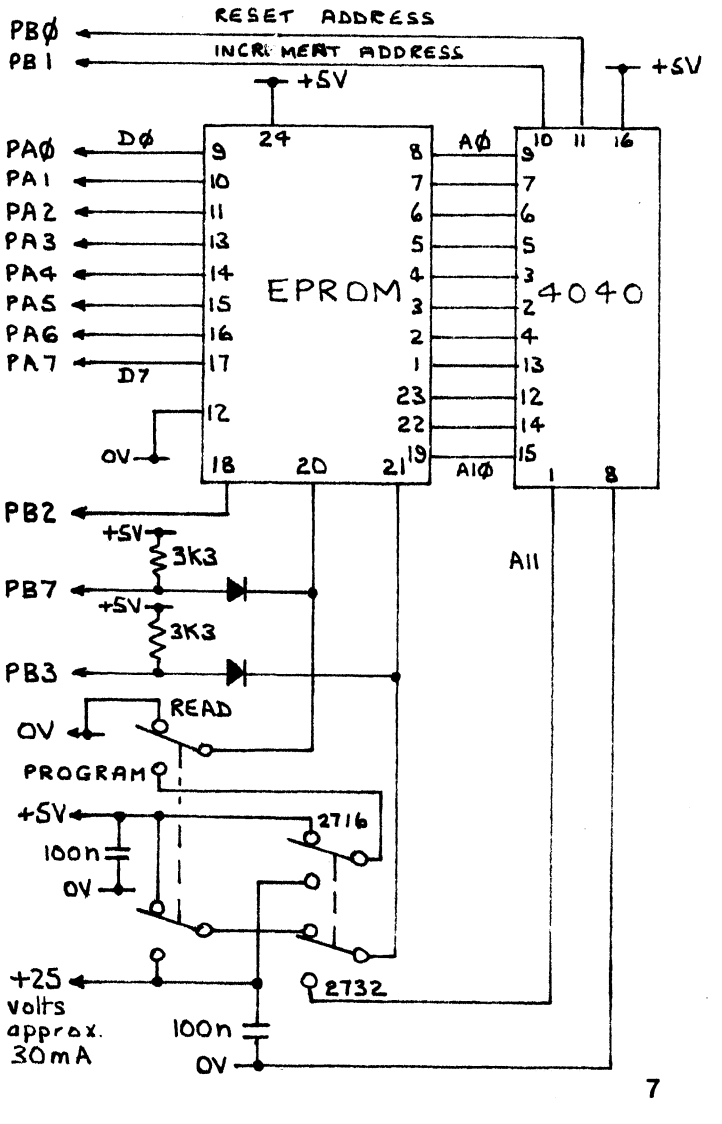
In the October "SUPERBOARD", we offered some modems. There are about 30 left and they are free, the only cost being registered postage to your address plus 60c for a handbook.

These modems accept RS232 levels, have 2kV opto-isolation between computer and line, features like DCD, RTS, and CTS, and originally cost several hundred dollars. They were used until recently by a large Australian corporation. Built in 1972, they do not use modern FSK techniques needed to access T.A.B. but are a reliable way to communicate from member to member.

If you are interested in obtaining one, write to me giving some details of your OSI system and what modifications and additions etc. that you have made. Include your telephone number and a SAE.

On January 31st, I will return your SAE's with details of costs for postage to your area. (Between $7 and $9.) Should I receive more requests than available modems, I will conduct a draw.

This offer is open to all KAOS members, as are other products mentioned in SUPERBOARD, such as the Character Sets, Eprom Extender board, Extended Basic Commands tape and BASIC 3 and BASIC 4 EPROMs.



SIMPLE EPROM PROGRAMMER Model 3
by John Whitehead

I have combined the circuits by Tony Vanbergen (KAOS Oct.81) and Brian Campbell (KAOS Jan.82) to enable 2716 and 2632 EPROMs to be programmed using a P.I.A.

I have written software for it in BASIC which can run at 1 MHz or 2 MHz and will:

Test for a clear EPROM.

Program any number of bytes, from a single byte to 2K or 4K.

Compare EPROM with memory and count any errors.

Move data from EPROM to RAM.

This program is available on tape for $2.00. Order at any meeting or by mail or phone for pick-up at the next meeting. If you want the tape posted, send $3.00 with your order.

AUTO BOOT AND MENU FOR PICO DOS

by Jeff Rae

Due to the interest shown in the Hidden File article in KAOS Vol 2 No 12, I have written two programs which will give you PICO DOS disks with 9 files and auto-boot of file 1.

The first program has to be entered with the OS65D3 DOS. This is the program which takes the tracks of the original PICO disk, modifies them and put them back on the disk. To use this program run it and follow the directions that are displayed on the screen. Please note the program as written expects you to insert a blank disk, so it will intialize the entire disk. If you are using your original PICO DOS disk and you don't want it wiped then answer N to the question "ARE YOU SURE?". This will cause the program to write only to tracks 0, 1 and 2. One final note this program assumes that you have an original PICO DOS disk that has the Track 0 R/W utility on track 2. If you are not sure that it has, the following operation will transfer it off OS65D3 to PICO DOS. Boot up OS65D3 and type in the following exactly:

```
                    EXIT(CR)
                    CA 4200=13,1(CR)
REMOVE OS65D3 DISK AND INSERT PICO DOS DISK
                    SA 02,1=4200/5(CR)
```

You now have the Track 0 R/W utility on Track 2.

After running program ONE you have a disk that will automatically LOAD and RUN the first file. Now, using PICO DOS type in the program called 'MENU' and SAVE it to File 1.

To use your new disk, type in your program or LOAD it from an existing disk and save it by SAVE(2-9), (you cannot use File 1 it is occupied by the MENU). Then to add your new program to the menu, LOAD1 (CR) RUN (CR) then press ESC. This will bring up the prompt DO YOU WISH TO CHANGE THE MENU? (Y/N) typing anything but Y will exit the program. An answer of Y will bring up a further prompt WHICH MENU ITEM DO YOU WISH TO CHANGE? Just type the number of the file and then the new name. If you wish to delete a name type " ". The name of the disk can be changed by typing 0 then the new name for the disk. You will find that you cannot change item 1 as the program assumes that it will always be MENU.

PROGRAM ONE

```
 10 B$=CHR$(10)+"   THEN PRESS ANY KEY"
 20 X=15:GOSUB300
 30 PRINT"This program when run and instructions
 followed will"
 40 PRINT"modify a PICO DOS Disk so that it will
 handle "
 50 PRINT"9 Files and auto boot in first file"
 60 PRINTB$:X=10:GOSUB300:GOSUB310
 70 A$="REMOVE DISK AND INSERT PICO DOS DISK":GO
SUB290
 90 GOSUB310
 100 DISK!"EX 41FD=00":DISK!"CA 4A00=01,1":DISK!
"CA 5200=02,1"
 110 A$="REMOVE PICO DOS DISK AND INSERT BLANK D
ISK":GOSUB290
 130 GOSUB310
 140 PRINT"BEWARE !!!! The next operation will o
verwrite the disk"
 150 PRINT"in the drive. Have you inserted the b
lank disk"
 160 DISK!"IN"
 170 DISK!"MEM 4E79,4E79"
 180 PRINT#5,"8955";CHR$(13);CHR$(13);CHR$(13);"
LOAD1";CHR$(13);
 190 PRINT#5,"POKE536,186";CHR$(58);"POKE537,255
";
 195 PRINT#5,CHR$(58);"RUN";CHR$(13);
 200 POKE17871,121:POKE17872,46:POKE17877,0:POKE
17848,9
 210 DISK!"SA 01,1=4A00/8":DISK!"SA 02,1=5200/5
"
 220 DISK!"MEM 5A00,5A00"
 230 PRINT#5,"EXIT";CHR$(13);"CA 0200=02,1";CHR$
(13);
 240 PRINT#5,"GO 0200";CHR$(13);"2";CHR$(13);
 250 PRINT#5,"W4200/2200,8";CHR$(13);"E";CHR$(13
);
 260 PRINT#5,"GO FF00";CHR$(13);
 270 DISK!"IO 10,02"
 280 END
 290 X=15:GOSUB300:PRINTA$:PRINTB$:X=15:GOSUB300
:RETURN
 300 FORY=1TOX:PRINT:NEXT:RETURN
 310 DISK!"GO 252B":RETURN
OK
```

```
 0 DATA"BLANK DEMO      "
 1 DATA"MENU            "
 2 DATA"                "
 3 DATA"                "
 4 DATA"                "
 5 DATA"                "
 6 DATA"                "
 7 DATA"                "
 8 DATA"                "
 9 DATA"                "
 50 PRINTCHR$(127)
 100 REM SET UP FOR C1-P OR C4-P
 105 IFPEEK(65506)=0THENL=32:C1=54016+PEEK(65504
):GOTO120
 110 L=64:C1=55040+PEEK(65504)
 120 WI=PEEK(65505):IFPEEK(57088)(127THENWI=31:P
OKE56832,0:FLAG=255
 130 C2=(C1-24*L)+INT((WI-23)/2)
 299 REM READ NAMES AND PUT ON SCREEN
 300 FORX=0TO9:READA$(X):NEXT
 310 FORX=0TO9:IFX()0THENPOKEC2+((X*2+3)*L)+3,X+
48
 350 FORY=1TOLEN(A$(X)):POKEC2+((X*2+3)*L)+5+Y,A
SC(MID$(A$(X),Y,1))
 360 NEXTY,X
 399 REM DRAWS BORDER AROUND MENU UNTIL KEY PRES
SED
 400 Z=161
 410 FORX=0TO23:POKEC2+X,Z:POKEC2+(23*L)+23-X,Z:
POKEC2+23+X*L,Z
 420 POKEC2+((23-X)*L),Z
 430 GOSUB500:REM SEE IF KEY PRESSED
 440 NEXT:Z=INT(RND(255)*254+1)
 450 GOTO410
 499 REM TEST TO SEE IF KEY PRESSED
 500 POKE530,1:KB=57088
 510 POKEKB,FLAG:P=PEEK(KB):IFFLAG=0THENP=255-P
 520 A=PAND254:POKE530,0:IFA=0THENRETURN:REM NO
KEY PRESS
 599 REM GET CHARACTER FOR KEY THEN LOAD AND RUN
 THAT FILE
 600 GOSUB800:IFA=27THENGOTO1000:REM GOTO CHANGE
 MENU IF ESC PRESSED
 610 POKE11908,A:POKE9679,121:POKE9680,46:PRINTC
HR$(127):POKE56832,1
 620 POKE536,206:POKE537,37 :END
 799 REM GET CHAR FROM KEYBOARD
 800 POKE11,0:POKE12,253:X=USR(X):A=PEEK(531):RE
TURN
 999 REM EXIT OR PUTS NAME IN DATA STATEMENTS AT
 0 TO 9
 1000 PRINTCHR$(127): POKE56832,1
 1005 PRINT"CHANGE MENU ? (Y/N)":GOSUB800:IFA()8
9THENEND
 1010 PRINT"WHICH MENU ITEM DO YOU WISH TO CHANG
E";:INPUTA
 1015 IFA(0ORA)9ORA=1THENGOTO1010
 1020 INPUT"NEW ITEM ";A$ :IFLEN(A$))14THENGOTO1
020
 1030 F=767:FORX=0TOA
 1035 IFPEEK(F)()131THENF=F+1:GOTO1035
 1039 F=F+1
 1040 NEXTX:FORX=1TOLEN(A$):POKEX+F,ASC(MID$(A$,
X,1)):NEXT
 1045 FORY=F+XTOF+14:POKEY,32:NEXTY
 1060 PRINT"ANY MORE CHANGES":GOSUB800:IFA=89THE
NGOTO1010
 1065 SAVE1
 1070 RUN
OK
```

Notes on the MENU program:

The maximum length of a file name is 14 characters. If the flashing border drives you crazy it can be disabled by 380 GOTO 600 or by deleting lines 399 to 520.

If you have any problems with these programs for a small copy charge you can get them from the disk library.

ASSEMBLER ENHANCEMENTS

by David Dodds

Users of the OSI assembler may be interested in some enhancements which are now available.

The first offering is an extension to the RESEQUENCE command written entirely in the psuedo code which comprises so much of the assembler. The new code allows user defined start and increment values to be set when resequencing. The command format is:

R(esequence)start value,increment value

Next page please

The routine first sets default options of 10 for both start and increment then checks for user values. Full syntax checking is carried out for each of the 4 legitimate combinations which may be used namely:

R	start=10	inc.=10
Rnnnn	start=nnnn	inc. =10
R,mmmm	start=10	inc. =nnnn
Rnnnn,mmmm	start=nnnn	inc. =mmmm

User values are initially checked to ensure that they do not exceed 65536 but no check is made for rollover during resequencing. The original routine did not have this facility and providing the test would have required a complete rewrite. No harm is done as the lines remain in the same place in memory as they were before resequencing (unless you save and reload text).

The new routine allows greater flexibility in that by numbering source code files with different values it is possible to concatenate them in a desired sequence. The catch is a loss of 48 bytes of precious memory!

About 12 months ago I added a modification to my assembler which was published in PEEK(65). The modification produces error messages instead of numbers. Investigation of the assembler has suggested a different method of detecting the occurrence of an error. By patching into the code which originates error messages and using a combination of psuedo code and machine code the error testing is made redundant. When an assembly error occurs the usual E# message is printed then a short form description of the error type. Approximately 1 page of memory is required to implement this mod.

I will make the source code for these routines available to any one who is interested. Send a stamped addressed envelope for print out only or $2 to cover cost of cassete and postage to David Dodds 9 Duke St, Ashburton, 3147 Vic. Complete instructions for implementing the modifications either for cassette or disk versions will be included.

At the time of writing further enhancements are being developed (such as autoline numbering). Any routines completed in the intervening period will also be included.

THE MEETING WAS KAOS

by Corky

It certainly was. The front row hecklers, (the Rabble Bored), were in fine form?!*?*! First, welcome to the three new members who made themselves known at the meeting and to the others who didn't.

Special guest, Bruce Coburn, gave a long talk and short demo of the Proton, the new BBC personal computer. We saw a very impressive display of colour graphics and were told all about why the BBC needed such a product, even if we didn't want to know. With due respect, Bruce is a very knowledgable and informed man, but he did tend to waffle-on about the less needed information and had to be cut short, leaving a lot more useful info unsaid.

Stewart Thomas, one of our young and very bright members, has another software mod to add to his growing collection, lower case BASIC that is recognised by the system. The first letter of each string is displayed as a capital and all others up to a carriage return or space, are lower case. The mod will even read a tape correctly and list to the screen in the new upper/lower format.

Next page please

Michael Lemaire, another very bright if noisy member, demonstrated what can be done with standard OSI graphics. The display appeared to be HI-RES but was in fact standard OSI resolution. Pictures of various well known space ships drawn with careful manipulation of the graphics set looked great.

Ray Gardiner had the RABBLE 65 (the new OSI compatible single board computer) up and running, great stuff Ray. The now very infamous Tony Durrant has his 8P up for sale and any reasonable offer, around $3500, will be considered , (I wish I had the money). Micom has a bulletin board running for those of you with modems. The cost is simply your phone call. The phone number is 03 762 5088 and the format is: 300 baud, 8 bits, no parity and no stop bit.

Joystick standard....(John Whitehead wrote this, not me)...Chief joystick designer got together with the Turtle man and the Tape maker, the three decided to standardize on 'C4P joystick B' for the club's single player games, (see KAOS April 81, July 81, Aug. 81). No agreement could be reached on 2 player games as the 'C4P joystick A' uses the number keys decoding. The club games are going to be modified for joystick use and the Tasan video board. The 'joystick B' is wired to the keyboard as:- W=fire, F=down, R=up, T=right, Y=left.

With GTBUG running successfully in the RABBLE 65 and the sudden appearance of a working version of George's assembler, I have no serial , saga nor tale to tell this month. What a shame, I was beginning to enjoy the stories myself. Well, until some time in 1983, from me, a very merry Christmas and a prosperous and fruitful (programming wise) New Year to you all.

DEAR PAUL, Christmas Edition

by Paul Dodd

Q. What are the 'Standard KAOS' I/O port locations?
A. NOTE: these addresses are from section 12, page 5 (I/O decoding) of the Rabble Ozi Expansion Board manual.

C000 - C003	Floppy disk PIA
C004 - C007	User PIA 1
C008 - C00B	User PIA 2
C00C - C00F	Calculator PIA (or User PIA 3)
C010 - C013	Floppy Disk ACIA
C014 - C017	Prog. Sound Generator PIA
C018 - C01B	Spare Cassette ACIA
C01C - C01F	Not allocated
C020 - C02F	Real Time Clock Clear
C030 - C03F	User VIA 1
C040 - C04F	User VIA 2

Some of these may be peculiar to the Rabble board, but they should give you a reasonable idea of the locations.
Q. Is there an extended BASIC EPROM available for the Superboard?
A. I have heard that the Queensland User Group (See the 'SUPERBOARD' section in this magazine) has an extended BASIC ROM called 'BASIC 5'. It would probably be advisible to contact Ed Richardson in Queensland (the address is on the 'SUPERBOARD' page) and get more information from him.

Next page please

Q. Is Pascal available for my 5.25" disk system with 24K RAM?
A. There are three Pascal implementations available. Firstly there is PASCELF which is a very tiny PASCAL that will run on anything from a C1P with 8K and cassette to a C8P with 48K and dual 8" disks. Secondly there is a mini-PASCAL which runs on a C1P or C4P with at least 32K and one 5.25" disk drive. Thirdly for the big systems, there is UCSD PASCAL which is a full implementation requiring 48K and dual 8" disks. (There is a 5.25" version, but I know of no-one in KAOS with a copy of it.)

Q. What are the track boundaries of BASIC, Assembler/Editor and Extended Monitor (ie. which track do they start on, which track do they finish on, and where are they loaded into RAM)? Is this information stored on disk anywhere?
A. Both BASIC and the Assembler-Extended Monitor combination are three tracks ie. 6K long, and both reside in memory from $0200 upwards. If you have an OS65D disassembly or better still, a source listing, you will notice that the 'ASM' command is located at $2ADE and looks like this:

```
        $2ADE A9 05      ASM LDA #$05   ; First track number
        $2AE0 20 EE 2A       JSR LDCMN  ; Common load routine
        $2AE3 4C 00 13       JMP STASM  ; Assembler entry point
```

The 'BASIC' and 'EM' commands are similar. The 'LDCMN' routine is at $2AEE, and expects to find a track number in the accumulator, it then loads three tracks, including the one specified, and stores them from $0200 upwards. Incidently, the 'BASIC' (or 'BA') command is at $2AE6 and 'EM' is at $2B2F. If you load the Assembler, you automatically load the Extended Monitor and vice versa.

Q. What are the numbers on the right of the screen when I run 'SCOPY'?
A. These numbers are supposed to represent the number of pages free after loading each track. They are correct for the first few tracks but there is a bug in the program and they are incorrect after the first disk swap.

Q. I would like to study disk BASIC. Is there a book similar to 'The First Book of OSI' (which is written for ROM machines) available for disk BASIC? Where else can I look for information?
A. There is not, as far as I know, a disk equivalent to the book you mentioned. There is no easy way to find out about all the routines but there are plenty of articles in 'PEEK 65' and 'The Aardvark Journal' outlining some of the functions of disk BASIC, and I think that there is a source listing of disk BASIC somewhere around.

Here are a couple of questions that I haven't been able to answer:

Q. On the standard system BASIC disk, there are about 30 bytes of code on track 6, sector 2 (5.25" systems). This code is something to do with the keyboard location ($DF00), but can apparently be deleted with no adverse effects. What is it for, and is it ever used?

Q. I have come across a problem when saving 8 pages of data on a track in more than one sector (ie. 2 sectors of 4 pages each). I can't do a 'DIR' on them (the system hangs), nor can I copy them. I've found (using the 'EXAM' command) that some of the data that should have gone onto the second sector was written to the space between the index hole and track start header. I've had partial success trying to correct this by changing location $270A (the delay between the index hole and the track header) from $0A to $09, can anyone shed any light on the problem?

I have a partial answer, but I would rather wait until I know the whole story before publishing it. If anyone has a real answer, could they please send it to me, and I will include it as part of a later 'Dear Paul' column.

I hope everyone has a Merry Christmas, and a very Happy New Year. I look forward to seeing you at the January Meeting.

THE BEGINNING MACHINE LANGUAGE PROGRAMMER....Part 6

by David Dodds

During the writing of a series like this the author gets to ask all sorts of self indulgent questions (like why did I volunteer for this anyway). The question every author dreads is "what did I forget?". Always you hope the answer is nothing.......Not so.....you have been mislead....Well almost!

This series is not really about machine language programming at all (so much for that title!). For the last few months you have really been learning Assembly Language. No one sane ever programs in machine language. After all a long string of binary numbers is very hard to understand. You have to learn about the machine instructions that assembly language symbolises in order to produce machine language programs, but even for learning these we use assembly language.

Assembly language programs are made up of numbered lines containing mnemonics and their associated addressing notation. Sometimes a line will also contain special symbols.

A complete program usually looks something like this:

```
 10 ;  Screen Clear
 20 ;
 30        *=$0222
 40 ;
 50 INIT   LDA #$00       ; Set up pointer to
 60        STA TVPUT+1    ; Top left corner of
 70        LDA #$D0       ; screen memory
 80        STA TVPUT+2
 90        LDA #$20       ; Blank chr.
100        LDX #$07       ; No of pages of mem.
110 TVPUT  STA $D000
120        INC TVPUT+1    ; set up next address
130        BNE TVPUT
140        INC TVPUT+2
150        DEX            ;page completed
160 ENDTST BPL TVPUT      ; branch if not finished
170        LDY #$00
180 TEXT   LDA MESAGE,Y
190        BEQ END
200        STA $D365,Y    ; for C1
210        INY
220        BPL TEXT
230 END                   ;END
240 MESAGE .BYTE 'READY',$00
```

Since there will be a lot more assembly language programs on these pages in future it would be appropriate to digress a little and cover some of the special symbols which you will be encountering.

Usually any places where values are stored are given names or labels such as TVPUT. In this way if you change the value or the location you don't have to change an entire program, just one line. This practise can also make programs far more readable; compare for example STA BUFFER,Y and STA $13,Y. Labels can also be used to identify branch entry points, tables, routines and subroutines either within or external to the program.

```
eg.  MOV.IT LDY #$10
            LDA (START),Y
            STA (END),Y
            DEY
            BPL MOV.IT+2
```

Next page please

Labels are usually a maximum of 6 characters long and start with an alpha character. The operator = (sometimes EQU) is used to assign values to labels eg START=$54.

Somewhere before the program (and sometimes during it as well) a line like *=$0222 appears. This means that the code which the program represents is to be placed from $0222 onwards. The * symbol is specifying the origin of the program.

Assembly language programs should always contain comments, otherwise next time you come to read them you won't have any idea what the code is doing. Comments are usually preceded by the ; symbol. Sometimes ; is used on its own to space out program segments and make the program more readable.

Values within programs can usually be expressed in a number of ways. Depending on the context, values might be expressed as ASCII characters, binary numbers, in decimal, octal or in hexadecimal. The machine code A9 43 could be derived from any of the following phrases in a program:

```
      LDA #'E          ASCII
   or LDA #$43         Hexadecimal
   or LDA #67          Decimal
   or LDA #%01000011   Binary
   or LDA #@103        Octal
```

The symbols ' $ % and @ indicate the type of value being represented. LDA #'E would clearly be more appropriate in a text or command mode situation while %01000011 might be used when preparing to test the condition of say an output port.

Arithmetic operators can also be used in an expression eg INC TVPUT+2 indicates that the location to be incremented is 2 locations past the position TVPUT. The operators +, -, / and * are all valid.

When it is necessary to include data such as constants or text it is usual to prefix this data using .WORD or .BYTE. These symbols are referred to as directives.

.WORD is used when an absolute address is to be placed in memory. The address is placed in standard 6502 low byte, high byte format. If only 2 digits are specified then a high order value of $0 is assumed.

.BYTE is used for single items of data. More than one byte may be specified on a line but successive values are separated by a comma. ASCII strings however may be given without the comma.

The completed program is usually referred to as a source listing. Another type of listing known as an assembly listing includes the start of memory at each line and the Hex characters which represent the machine language produced by that line. This additional information is usually placed between the line number and the area used for labels. Since the machine code is listed on the assembly listing it can be used to enter the code into your machine by hand if you do not have an assembler.

Future listings will be source listings. I suggest that at first you hand code them and use the monitor to enter them in your machine. Since I will be endeavouring to teach programming techniques as well as assembly language the way future programs are presented might not necessarily be the best way to write them. Try experimenting with alternate methods of achieving the same result. It doesn't matter if it bombs you won't blow up you computer. The best way to learn is by your mistakes.

USER-FRIENDLY PROGRAMMING
by Michael Lemaire

Microcomputers are becoming increasingly important to people nowadays. Business people want to keep information where they can get at it quickly, to have their whole business at their finger tips. At home, people want to keep up with the Jones by having a trendy microcomputer; but they don't want to have to be programmers to use it, and the people in business probably don't have the time or inclination to become programmers. All this means that microcomputers have a very comfortable (and lucrative) niche in society -- as APPLIANCES, rather than as 'great powerful microcomputers' with 64K RAM and dual disk'. This means that they must be easily used as appliances, rather than needing a computer background to operate them. This is why ATARI has been so successful -- their programs don't rely on the user knowing what a byte or a sector is, themost knowledge you need to run an Atari software package, whether it is a game or inventory system, is how to push the start button. As a result, anyone can use it, and the market for the machine and software is huge.

Recently I was asked to write a stock control system for a company with no previous computer background; thus the system needed to be as friendly as possible for the people using it. I came to a few conclusions while designing the system;

(1) Write the system so that everything is obvious and all the information you need to know at a given point is on the screen in front of you. Not that you don't supply documentation; just that the user should not have to be forced to spend a lot of time learning the system or referring to the documentation.

(2) Make it impossible to input a bad command. If the user gets an error message every time he enters a bad command he may get a little peeved or feel inferior. In the system I wrote, all commands are entered by 'pointing' to a menu choice with an inverse-video cursor. The user can only enter a menu choice; nothing else which could give an error. The concept of an illegal command does not exist. This was one reason why I chose to have commands entered by a menu system rather than a command-language style of command entry.

(3) Arrange the system logically. The system should, if possible, copy 'real life' situations. The user will feel more at home with something that makes sense and works the way he feels it should. My stock control system is arranged so the user decides what he wants to do (eg. look up records, enter a new record). He then enters what record he wants to operate on; item records, or supplier records, say. A record is displayed on the screen set up like a card file (yes, just like 'real life). When a field is changed, the new contents are drawn up on the 'card'.

(4) Be consistent. The user will get confused if one method of eg. command input, is used in one section of the package, and a different system used somewhere else. With a consistent enviroment, the user will be able to deduce how to use an unfamiliar section, because it will work in a similar fashion to the other sections he has used.

This sort of user-friendly operation is not restricted to business packages. Other things, like games need them. There is nothing worse than starting up a game or word processor or whatever on my OSI and having no idea what to do, or what keys to press. But there are also many friendly programs around which are a joy to use. Byte Vol.7 No.4 (April 1982) has a lot of good information on this subject.

(By the way, is anybody interested in an inventory system or stock-control program with point-of-sale and automatic stock update?)
Contact Michael by writing C/O KAOS.

GOOD NEWS FOR OSI OWNERS
by Michael Lemaire

If you looked at the benchmark summary in the November issue of APC, then you may have been a little upset by the C4P's apparently poor showing with a total run time of 25.0 seconds. Well, I tried out the benchmarks, and decided that they were run on a 1MHz machine. So if you have a 2MHz system, then your benchmark time is in fact 12.5 seconds, putting OSI second after the Olivetti (and faster than the BBC Micro!). It goes to show that newer is not always faster.

Olivetti M20..................	11.5 sec. average.
** OSI C4P **.................	12.5
BBC Micro.....................	14.6
IBM Personal Computer.........	17.6
Sirius 1......................	24.8
VIC 20........................	28.7
Hitachi Peach.................	39.5

On the individual benchmarks, the 2MHz OSI comes first on numbers 3, 6, and 7. We should all give thanks to Richard W. Weiland for making KAOS such an elite group of micro users.

THE BBC MICROCOMPUTER
by Jeff Rae

At the November meeting a BBC Proton was demonstrated by Brian Coburn from BARSON COMPUTERS. Unfortunatly the demonstration was cut short due to the lack of time but we were treated to a brief history of the ACORN systems in general and the development of the PROTON.

The PROTON on display was a model B which features a 2 Mhz 6502A micro, 32K of RAM, 32k of ROM, cassette interface at 300 or 1200 baud, RS423 serial port, parallel ports for printers etc, high resolution colour graphics and many other features that add up to the very attactive package.

We were also shown a range of the games software. The games demonstrated were a version of Ghost Muncher called Snapper and Defender, all of the software made extensive use of the high resolution graphics and sound to produce a game as good as the original arcade games. Other software available includes Forth, Lisp, Pascal, and a lot of educational programs.

For pricing and further details contact:
BARSON COMPUTERS 86 Nicholson St. Abbotsford, VIC, 3067 Phone 03 419 3033

FOR SALE OR SWAP

C4P - 8K, 300/600 baud, 1-2 MHz. Manuals and lots of programs including Cursor Control, Editor/Assembler, WP6502 and many games - most home grown.
NTSC colour, sound. $550.00. B & W converted TV $50.00.
Write to Peter Meulman,

FOR SALE C8P $3500 (negotiable). Contact Tony Durrant on 03 428 4108 AH.

24K Static Ram Memory Board, OSI 540 board. Has run without fault. Sell for $225 or swap for OSI compatable 5.25" or 8" floppy disk drive.
David Wilson

I have a copy of OS65U V1.3 CD-74/CD-36 on 8" floppy to swap. I am interested in swapping it for a Tasker Buss kit - preferable memory board, I/O board or EPROM board. The floppy comes with the OS65U Reference manual.
Contact Derryl Cocks,

OOPS

There was a typing error in the Ext Mon Formatted Output program on page 3 of KAOS Vol 3 No 2. 1F58 should read 20AF1B and line 1F5B should be CA